#3

ART 34 AMDT

Patent Claims

1. Method for protecting several programs and/or several files from unauthorized access by a process,

- in which each program and/or each file to be protected is assigned an address space,

- in which each program and/or each file to be protected is also assigned a process file,

- in which the process or processes that may run in the address space in question is or are stored in a process file,

- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,

- in which the cryptographic value of one process is contained in the process file,

- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,

- in which the accessing process's cryptographic value is formed, and

- in which the cryptographic values of the processes are compared with each other during the check,

- in which, if the accessing process is included in the process file, the accessing process is started, and

- otherwise, the accessing process is not started.

2. Method for protecting several programs and/or several files from unauthorized access by a process,

- in which each program and/or each file to be protected is assigned an address space,

- in which each program and/or each file to be protected is also assigned a process file,

- in which the process or processes that may run in the address space in question is or are stored in a process file,

- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,

- in which the cryptographic value of one process is contained in the process file,

- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,

- in which the accessing process's cryptographic value is formed, and

- in which the cryptographic values of the processes are compared with each other during the check,

- in which, if the accessing process is included in the process file, the accessing process is continued, and

- otherwise, the accessing process is ended.

3. Method according to one of the Claims 1 or 3 [sic],

in which in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process is forwarded to a checking function in which the check is carried out.

4. Method according to Claim 3,

in which the checking function can be integrated into the address space of the program and/or the file to be protected as a dynamically integrated file.

5. Method according to Claim 1 or 2,

- in which a call of the accessing process is forwarded to a checking function in which the check takes place, and

-in which the checking function is integrated into an operating system core of an operating system with which the programs are executed.

6. Method according to one of the Claims 1 through 5,
in which the operating system is Windows NT.

7. Method according to one of the Claims 1 through 6,
in which at a predetermined interval of time for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program and/or the file to be protected, and the process is ended if that is not the case.

8. Method according to one of the Claims 1 through 7,
in which in dependency on a predetermined event for each active process that runs along with a program and/or a file to be protected, a check is made to confirm whether the active process is contained in the process file that is assigned to the program to be protected, and the process is ended if that is not the case.

9. Method according to one of the Claims 1 through 8,
in which a protection program for protecting the method according to one of the preceding Claims can be executed is stored encoded and is decoded at the start of the method.

10. Method according to one of the Claims 1 through 9,
in which the programs and/or the files to be protected are stored encoded and are decoded at the

start of the method according to one of the preceding Claims.

11. Method according to Claim 9,

in which after the decoding of the protection program, its integrity is checked and the method according to one of the preceding Claims is executed only if the integrity of the protection program is assured.

12. Method according to Claim 11,

in which after the integrity test of the protection program, the integrity of all processes contained in the process files is checked and the method according to one of the preceding Claims is executed only if the integrity of all of the processes contained in the process files is assured.

13. Method according to Claim 12,

in which after the integrity test of the processes, the integrity of the program and/or the file to be protected is checked and the method according to one of the preceding Claims is executed only if the integrity of the program and/or the file to be protected is assured.

14. Method according to Claim 12 or 13,

in which at least one of the integrity tests takes place through the use of a cryptographic method.

15. Method according to one of the Claims 1 through 14,

in which the cryptographic value is formed through the use of a hash function.

16. Array for protecting several programs from unauthorized access by a process,

with a processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,

- a process file is assigned to each program and/or each file to be protected,

- the process or processes that may run in the address space in question is or are stored in a process file,

- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,

- in which the cryptographic value of one process is contained in the process file,

- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,

- in which the accessing process's cryptographic value is formed, and

- in which the cryptographic values of the processes are compared with each other during the check,

- in which if the accessing process is included in the process file, the accessing process is started, and

- otherwise, the accessing process is not started.

17. Array for protecting several programs from unauthorized access by a process,

with a processor that is set up in a way such that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,

- a process file is assigned to each program and/or each file to be protected,

- the process or processes that may run in the address space in question is or are stored in a process file,

- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,

- in which the cryptographic value of one process is contained in the process file,

- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,

- in which the accessing process's cryptographic value is formed,

- in which the cryptographic values of the processes are compared with each other during the check,

- in which if the accessing process is included in the process file, the accessing process is continued, and

- otherwise the accessing process is ended.

18. Array according to one of the Claims 16 or 17,
in which the processor is set up in such a way that in a call mechanism for a function of an operating system core with which the programs are executed, a call of the accessing process is forwarded to a checking function in which the check is carried out.

19. Array according to one of the Claims 16 through 18,
in which the processor is set up in such a way that the operating system is Windows NT.

20. Set of several arrays and a server array which is connected with each array of the set of

several arrays and which is to protect several programs from unauthorized access by a process, whereby each array exhibits a processor that is set up in such a way that the following steps can be carried out:

- an address space is assigned to each program and/or each file to be protected,
- a process file is assigned to each program and/or each file to be protected,
- the process or processes that may run in the address space in question is or are stored in a process file,
- in which for at least a part of the processes included in a process file, a cryptographic value that uniquely identifies the process is formed,
- in which the cryptographic value of one process is contained in the process file,
- in which, during the running of a program and/or a processing of a file to be protected, for a process that attempts to access the address space of the program and/or the file to be protected, a check is made to confirm whether the accessing process is included in the corresponding process file,
- in which the accessing process's cryptographic value is formed, and
- in which the cryptographic values of the processes are compared with each other during the check,
- in which if the accessing process is included in the process file, the accessing process is started or continued, and
- otherwise an alarm signal is generated and sent to the server array,

and whereby the server array exhibits a processor that is set up in such a way that a predetermined action is triggered in dependency on at least one received alarm signal.